Atlas Scientific™
Environmental Robotics

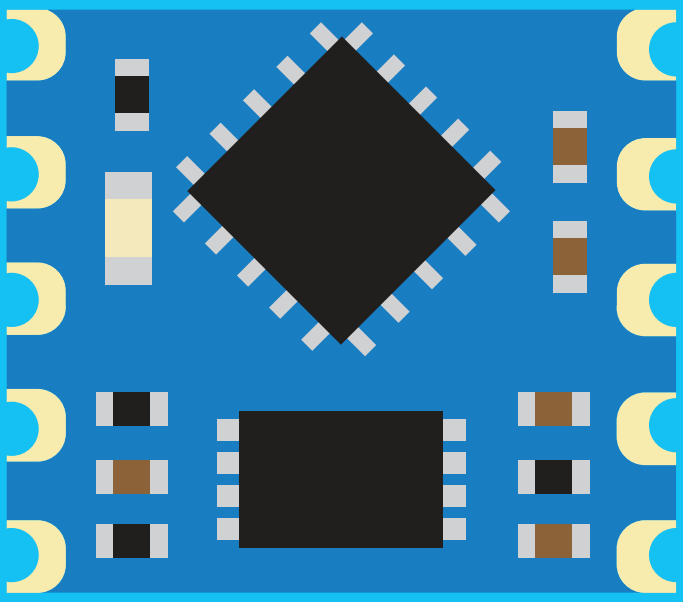
V 3.1

Revised 11/24

OEM-ORP™

Embedded ORP Circuit

Reads	**ORP**
Range	**-1020mV – 1020mV**
Accuracy	**+/– 1mV**
Response time	**1 reading every 420ms**
Supported probes	**Any type & brand**
Calibration	**Single point**
Temp compensation	**N/A**
Data protocol	**SMBus/I²C**
Default I²C address	**0x66**
Operating voltage	**3.0V – 5.5V**
Data format	**ASCII**



RoHS COMPLIANT

PATENT PROTECTED

Written by Jordan Press
Designed by Noah Press

This is an evolving document, check back for updates.

STOP

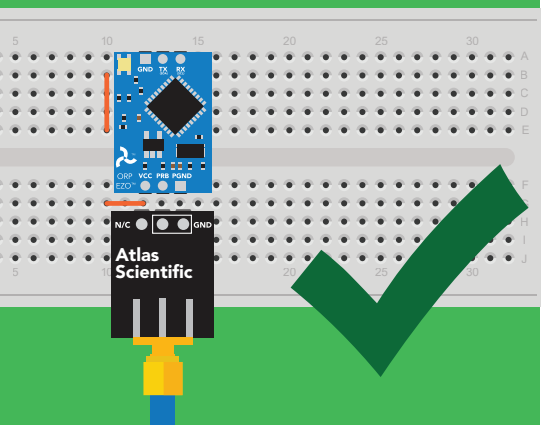
SOLDERING THIS DEVICE VOIDS YOUR WARRANTY.

Before purchasing the ORP OEM™ read this data sheet in its entirety. This product is designed to be surface mounted to a PCB of your own design.

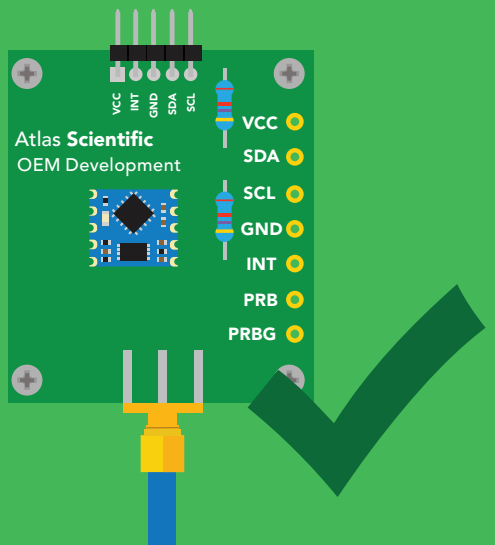
This device is designed for electrical engineers who are familiar with embedded systems design and programing. If you, or your engineering team are not familiar with embedded systems design and programing, Atlas Scientific does not recommend buying this product.

Unfamiliar with ORP sensing? Try our EZO™ ORP circuit first.

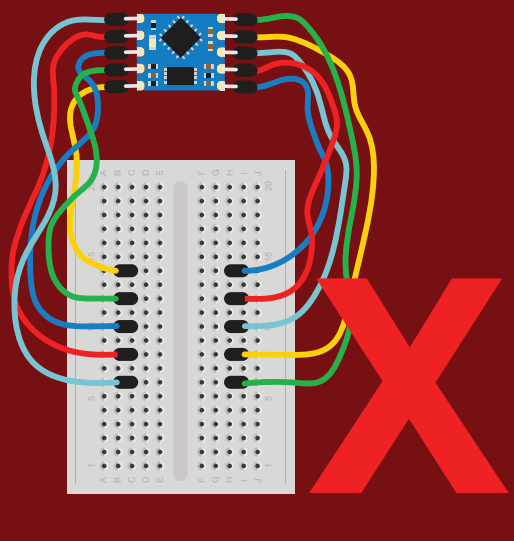
It's much easier to use, and provides a good working reference.

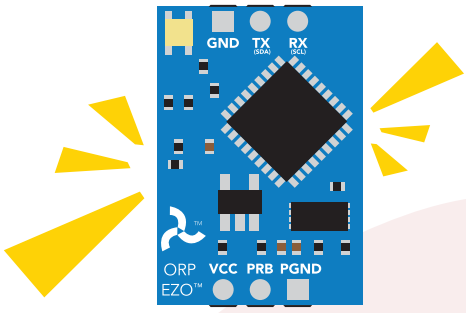


Get this device working in our OEM Development board first!



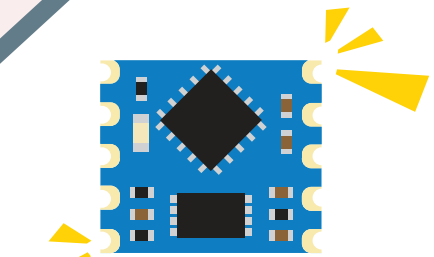
Do not solder wires to this device.





EZO-ORP

Getting an EZO-ORP circuit working first will significantly speed up OEM development. It will act as a working reference model; making it easy to debug OEM issues.



ORP-OEM

Because an EZO-ORP circuit was used first, product development time was reduced by 3 weeks.

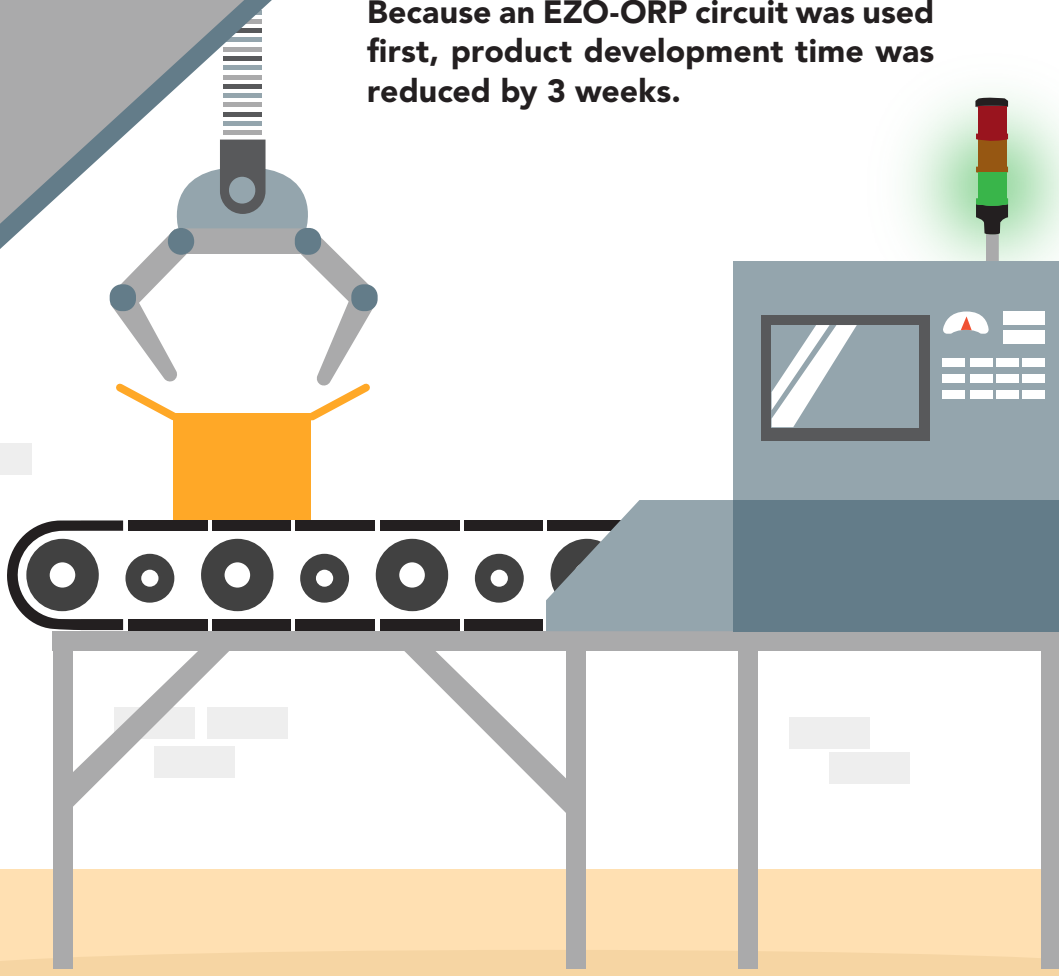
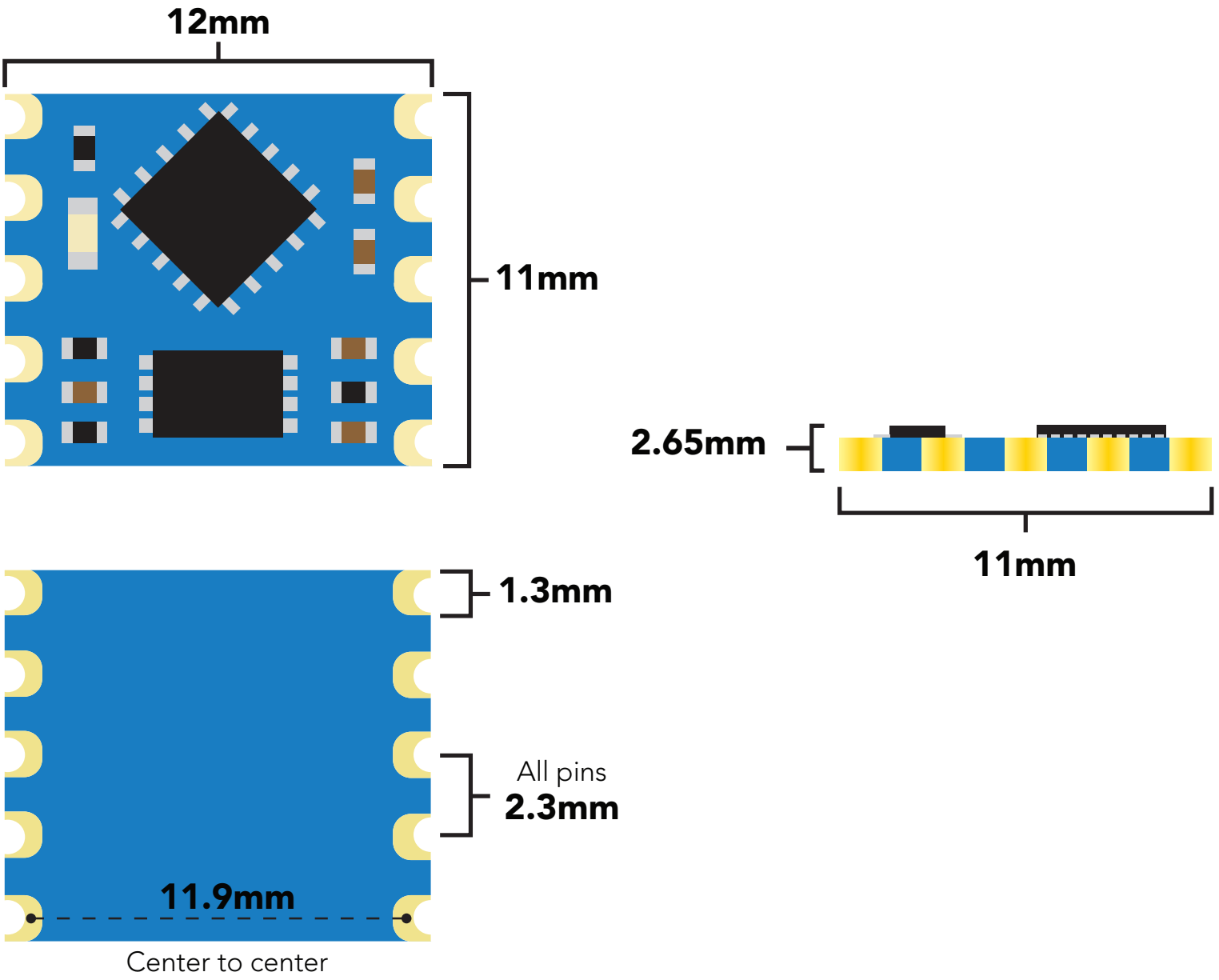


Table of contents

REGISTERS

OEM circuit dimensions



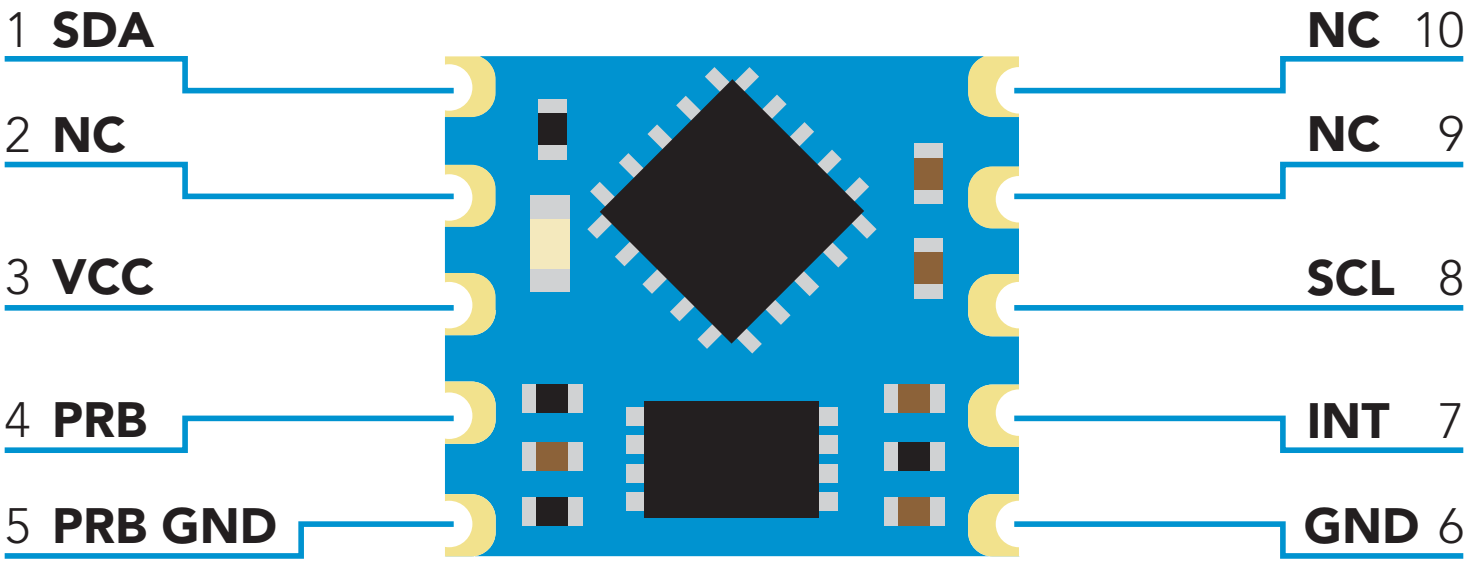
Power consumption

	LED	OPERATIONAL	HIBERNATION
3.3V	ON	3.46 mA	3.43 mA
	OFF	3.03 mA	3.0 mA

Absolute max ratings

Parameter	MIN	TYP	MAX
Storage temperature	-60 °C		150 °C
Operational temperature	-40 °C	25 °C	125 °C
VCC	3.0V	3.3V	5.5V

Pin out



Resolution

The resolution of a sensor is the smallest change it can detect in the quantity that it is measuring. The Atlas Scientific™ ORP OEM™ will always produce a reading with a resolution of one decimal place.

Example
1.2 mV
-190.6 mV

Power on/start up

Once the Atlas Scientific™ ORP OEM™ is powered on it will be ready to receive commands and take readings after 1ms. Communication is done using the SMBus/I²C protocol at speeds of 10 – 100 kHz.

Settings that are retained if power is cut

Calibration
I²C address

Settings that are *NOT* retained if power is cut

Active/Hibernation mode
LED control
Interrupt control

System overview

The Atlas Scientific ORP OEM™ Class Embedded ORP Circuit is the core electronics needed to read the ORP of water from any off the shelf ORP probe. The ORP OEM™ Embedded Circuit will meet, or exceed the capabilities and accuracy found in all models of bench top laboratory grade ORP meters.

The ORP OEM™ is an SMBus / I²C slave device that communicates to a master device at a speed of 10 – 100 kHz. Read and write operations are done by accessing **18** different 8 bit registers.

Accessible registers

Device information

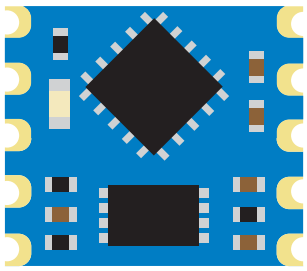
0x00: Device type	R
0x01: Firmware version	R

Device address

0x02: SMBus/I²C address lock/unlock	R/W
0x03: SMBus/I²C address	R/W

Control

0x04: Interrupt control	R/W
0x05: LED control	R/W
0x06: Active/hibernate	R/W
0x07: New reading available	R/W



Read only R
Read and write R/W

Calibration

0x08: Calibration value MSB	R/W
0x09: Calibration value high byte	R/W
0x0A: Calibration value low byte	R/W
0x0B: Calibration value LSB	R/W

0x0C: Calibrate request	R/W
0x0D: Calibration confirm	R

Sensor Data

0x0E: ORP reading MSB	R/W
0x0F: ORP reading high byte	R/W
0x10: ORP reading low byte	R/W
0x11: ORP reading LSB	R/W

The default device address is **0x66**
This address can be changed.

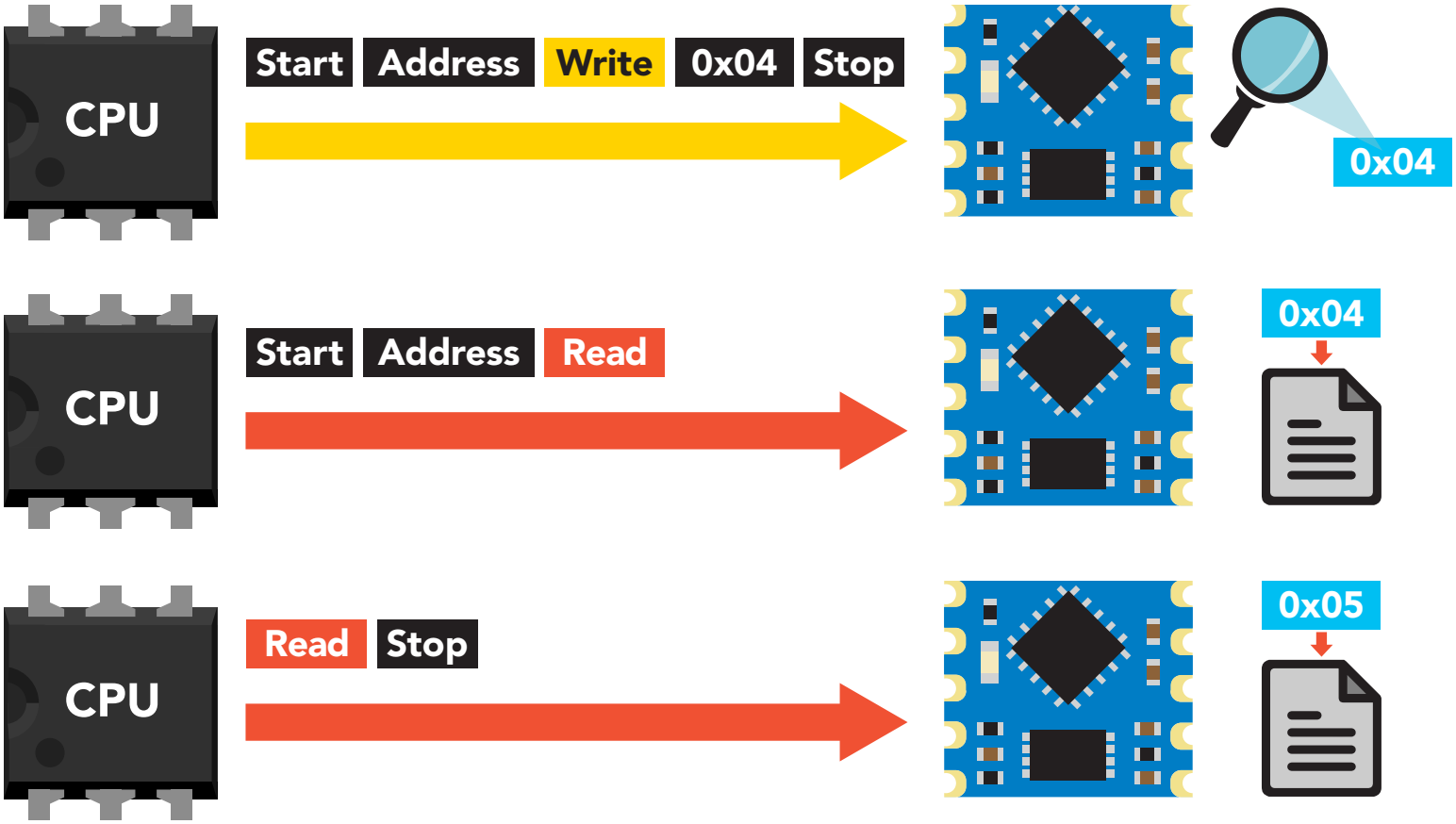
Each ORP reading takes 420ms

Reading register values

To read one or more registers, issue a write command and transmit the register address that should be read from, followed by a stop command. Then issue a read command, the data read will be the value that is stored in that register. Issuing another read command will automatically read the value in the next register. This can go on until all registers have been read. After reading the last register, additional read commands will return 0xFF. Issuing a stop command will terminate the read event.

The default device address is **0x66**
This address can be changed.

Example
Start reading at register 0x04 and read 2 times.



Example code
reading two registers

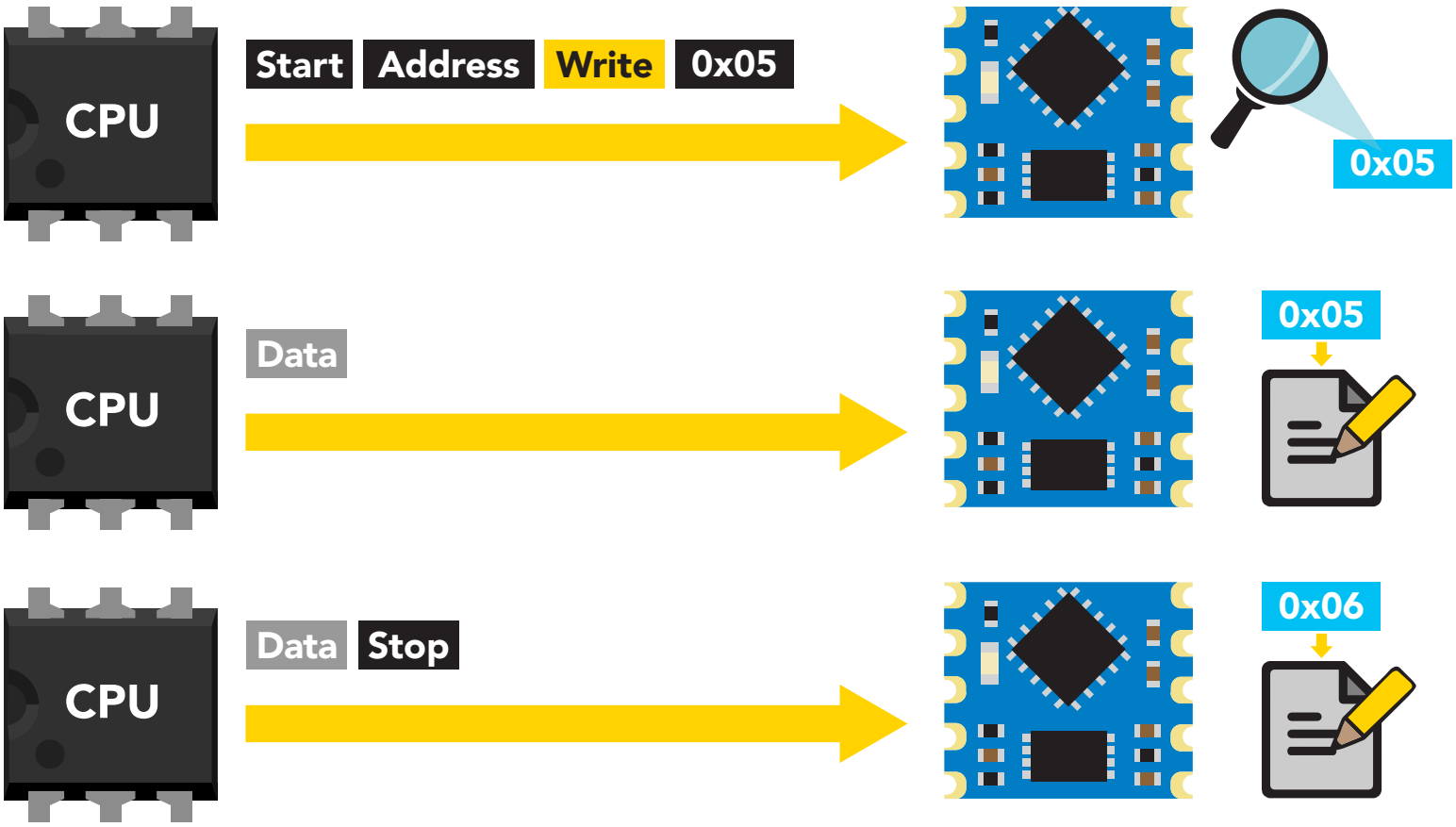
```
byte i2c_device_address=0x66;
byte reg_4, reg_5;

Wire.beginTransmission(i2c_device_address);
Wire.write(0x04);
Wire.endTransmission();

Wire.requestFrom(i2c_device_address,2);
reg_4=Wire.read();
reg_5=Wire.read();
Wire.endTransmission();
```

Writing register values

All registers can be read, but only registers marked read/write can be written to.

To write to one *(or more)* registers, issue a write command and transmit the register address that should be written to, followed by the data byte to be written. Issuing another write command will automatically write the value in the next register. This can go on until all registers have been written to. After writing to the last register, additional write commands will do nothing.

Example
Start writing at address 0x05 and write 2 values.



Example code
writing the number 1 in register 0x05 – 0x06

```
byte i2c_device_address=0x66;
byte starting_register=0x05
byte data=1;

Wire.beginTransmission(i2c_device_address);
Wire.write(starting_register);
Wire.write(data);
Wire.write(data);
Wire.endTransmission();
```

Sending floating point numbers

For ease of understanding we are calling fixed decimal numbers "floating point numbers." We are aware they are not technically floating point numbers.

It is not possible to send/receive a floating (fixed decimal) point number over the SMBus/ I^2C data protocol. Therefore, a multiplier/divider is used to remove the decimal point. Do not transmit a floating point number without property formatting the number first.

When transmitting a floating point number to the calibration value registers, the number must first be multiplied by 10. This would have the effect of removing the floating point. Internally the ORP OEM™ will divide the number by 10, converting it back into a floating point number.

Example
Setting an ORP calibration value of: 125.6
125.6 X **10** = 1256
Transmit the number 1256 to the Calibration value registers

Setting an ORP calibration value of: -80
-80 X **10** = -800
Transmit the number -800 to the Calibration value registers

When reading back a value stored in the calibration value registers, the value must be divided by 10 to return it to its originally intended value.

Receiving floating point numbers

After receiving a value from the ORP reading registers, the number must be divided by 10 to convert it back into a floating point number.

Example
Reading an ORP value of 14.5
Value received = 145
145 / **10** = 14.5

Reading an ORP value of -281.3
Value received = -2813
-2813 / **10** = -281.3

Registers

Device information

0x00 – Device type register

1 unsigned byte
Read only value = 2
2 = ORP

This register contains a number indicating what type of OEM device it is.

0x01 – Firmware version register

1 unsigned byte
Read only value = 2
2 = firmware version

This register contains a number indicating the firmware version of the OEM device.

Example code
reading device type and device version registers

```
byte i2c_device_address=0x66;
byte starting_register=0x00
byte device_type;
byte version_number;

Wire.beginTransmission(i2c_device_address);
Wire.write(staring_register);
Wire.endTransmission();

Wire.requestFrom(i2c_device_address,(byte)2);
device_type = Wire.read();
version_number = Wire.read();
Wire.endTransmission();
```

Changing I²C address

0x02: SMBus/I²C address lock/unlock	R/W
0x03: SMBus/I²C address	R/W

This is a 2 step procedure

To change the I²C address, an unlock command must first be issued.

Step 1
Issue unlock command

0x02 – I²C address unlock register

1 unsigned byte
Read only value = 0 or 1
0 = **unlocked**
1 = **locked**

To unlock this register it must be written to twice.

Start	unlock register	0x55	Stop
Start	unlock register	0xAA	Stop

The two unlock commands must be sent back to back in immediate succession. No other write, or read event can occur. Once the register is unlocked it will equal 0x00 (unlocked).

To lock the register
Write any value to the register other than 0x55;
or, change the address in the Device Address Register.

Example code
address unlock

```
byte i2c_device_address=0x66;
byte unlock_register=0x02;

Wire.beginTransmission(bus_address);
Wire.write(unlock_register);
Wire.write(0x55);
Wire.endTransmission();

Wire.beginTransmission(bus_address);
Wire.write(unlock_register);
Wire.write(0xAA);
Wire.endTransmission();
```


Step 2
Change address

0x03 – I²C address register

1 unsigned byte
Default value = **0x66**
Address can be changed **0x01 – 0x7F (1–127)**

Address changes outside of the possible range **0x01 – 0x7F (1–127)** will be ignored.

After a new address has been sent to the device the Address lock/unlock register will lock and the new address will take hold. It will no longer be possible to communicate with the device using the old address.

Settings to this register are retained if the power is cut.

Example code
changing device address

```
byte i2c_device_address=0x66;
byte new_i2c_device_address=0x60;
byte address_reg=0x03;

Wire.beginTransmission(bus_address);
Wire.write(address_reg);
Wire.write(new_i2c_device_address);
Wire.endTransmission();
```


Control registers

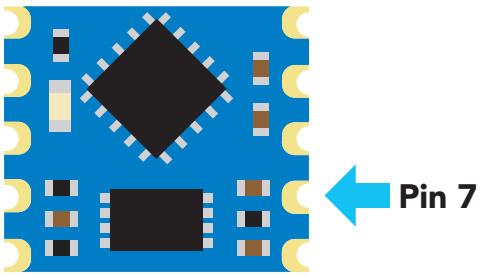
0x04: Interrupt control	R/W
0x05: LED control	R/W
0x06: Active/hibernate	R/W
0x07: New reading available	R/W

0x00
0x01
0x02
0x03
0x04
0x05
0x06
0x07
0x08
0x09
0x0A
0x0B
0x0C
0x0D
0x0E
0x0F
0x10
0x11

0x04 – Interrupt control register

1 unsigned byte
Default value = 0 (disabled)

Command values
0 = disabled
2 = pin high on new reading (manually reset)
4 = pin low on new reading (manually reset)
8 = invert state on new reading (automatically reset)



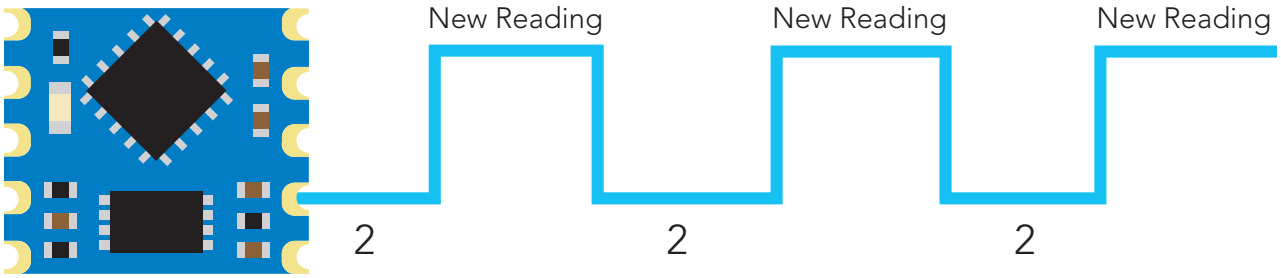
The Interrupt control register adjusts the function of pin 7 (the interrupt output pin).

Settings to this register are **not** retained if the power is cut.

Pin high on new reading

Command value = 2

By setting the interrupt control register to 2 the pin will go to a low state (0 volts). Each time a new reading is available the INT pin (pin 7) will be set and output the same voltage that is on the VCC pin.



The pin will not auto reset. 2 must be written to the interrupt control register after each transition from low to high.

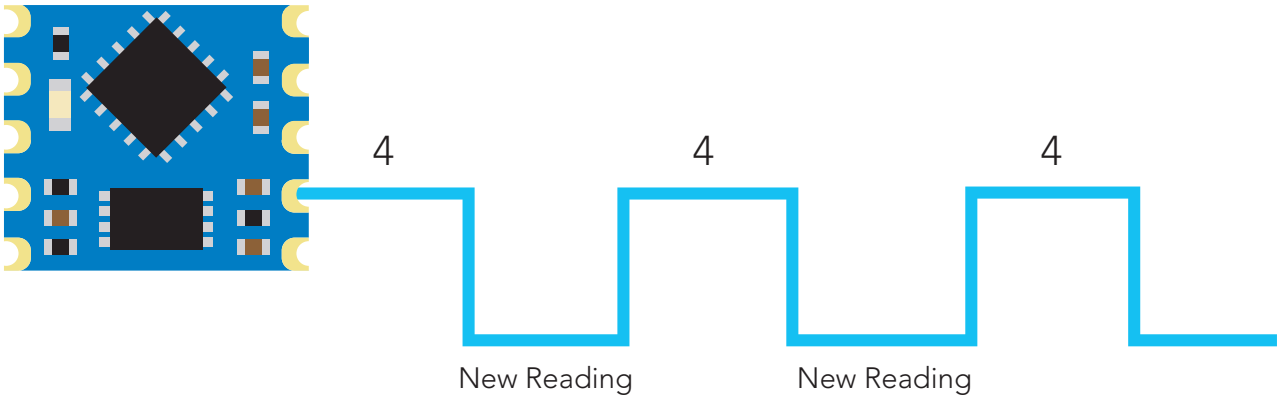
Example code
Setting pin high on new reading

```
byte i2c_device_address=0x66;
byte int_control=0x04;

Wire.beginTransmission(i2c_device_address);
Wire.write(int_control);
Wire.write(0x02);
Wire.endTransmission();
```


Pin low on new reading

Command value = 4

By setting the interrupt control register to 4 the pin will go to a high state (VCC). Each time a new reading is available the INT pin (pin 7) will be reset and the pin will be at 0 volts.



The pin will not auto set. 4 must be written to the interrupt control register after each transition from high to low.

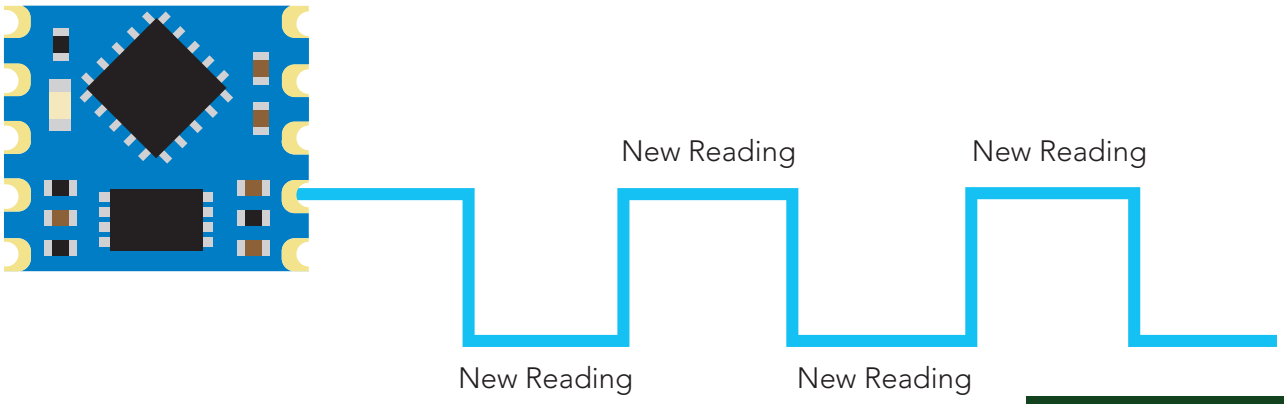
Example code
Setting pin low on new reading

```
byte I2C_device_address=0x66;
byte int_control=0x04;

Wire.beginTransmission(I2C_device_address);
Wire.write(int_control);
Wire.write(0x04);
Wire.endTransmission();
```

Invert state on new reading

Command value = 8

By setting the interrupt control register to 8 the pin will remain in whatever state it is in. Each time a new reading is available the INT pin (pin 7) will invert its state.



The pin will automatically invert its state each time a new reading is available. This setting has been specifically designed for a master device that can use an interrupt on change function.

Example code
Inverting state on new reading

```
byte i2c_device_address=0x66;
byte int_control=0x04;

Wire.beginTransmission(i2c_device_address);
Wire.write(int_control);
Wire.write(0x08);
Wire.endTransmission();
```


0x05 – LED control register

1 unsigned byte

Command values
1 = Blink each time a reading is taken
0 = Off

The LED control register adjusts the function of the on board LED. By default the LED is set to blink each time a reading is taken.

Example code
Turning off LED

```
byte i2c_device_address=0x66;
byte led_reg=0x05;

Wire.beginTransmission(i2c_device_address);
Wire.write(led_reg);
Wire.write(0x00);
Wire.endTransmission();
```

Settings to this register are **not** retained if the power is cut.

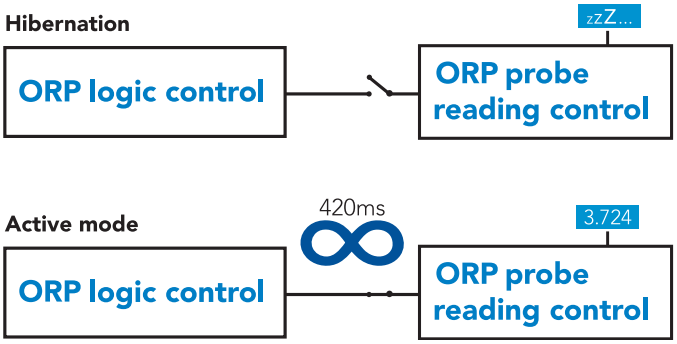
0x06 – Active/hibernate register

1 unsigned byte

To wake the device
Transmit a 0x01 to register 0x06

To hibernate the device
Transmit a 0x00 to register 0x06

This register is used to activate, or hibernate the sensing subsystem of the OEM device.



Example code
Activate ORP readings

```
byte i2c_device_address=0x66;
byte active_reg=0x06;

Wire.beginTransmission(i2c_device_address);
Wire.write(active_reg);
Wire.write(0x01);
Wire.endTransmission();
```

Once the device has been woken up it will continuously take readings every 420ms. **Waking the device is the only way to take a reading. Hibernating the device is the only way to stop taking readings.**

0x07 – New reading available register

1 unsigned byte
Default value = 0 (no new reading)
New reading available = 1

Command values
0 = reset register

This register is for applications where the interrupt output pin cannot be used and continuously polling the device would be the preferred method of identifying when a new reading is available.

When the device is powered on, the New Reading Available Register will equal 0. Once the device is placed into active mode and a reading has been taken, the New Reading Available Register will move from 0 to 1.

This register will never automatically reset itself to 0. The master must reset the register back to 0 each time.

Example code
Polling new reading available register

```
byte i2c_device_address=0x66;
byte new_reading_available=0;
byte nra=0x07;

while(new_reading_available==0){
Wire.beginTransmission(i2c_device_address);
Wire.write(nra);
Wire.endTransmission();

Wire.requestFrom(i2c_device_address,(byte)1);
new_reading_available = Wire.read();
Wire.endTransmission();
delay(10);
}

if(new_reading_available==1){
call read_ORP();
Wire.beginTransmission(i2c_device_address);
Wire.write(nra);
Wire.write(0x00);
Wire.endTransmission();
}
```


Calibration

0x08: Calibration value MSB	R/W
0x09: Calibration value high byte	R/W
0x0A: Calibration value low byte	R/W
0x0B: Calibration value LSB	R/W

0x08 – 0x0B Calibration registers

Signed long
0x08 = MSB
0x0B = LSB
Units = mV

A calibration point can be a singed whole number, or singed floating point number with one decimal place.

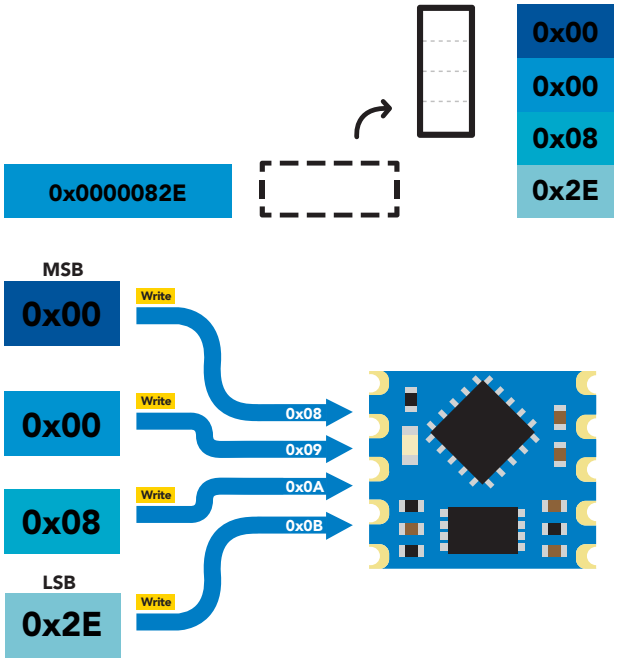
Example
225.0
-12.5

After sending a value to this register block, calibration is **not** complete. The calibration request register must be set after loading a calibration value into this register block.

To send a new calibration value to the ORP OEM™ the value of the calibration solution must be multiplied by 10 and then transmitted to the ORP OEM™. *(The calibration value will be divided by 10 internally).* Move the value from a float to a signed long. Break up the signed long into its 4 individual bytes. Send the bytes (MSB to LSB) to registers 0x08, 0x09, 0x0A and 0x0B.

Example
calibrating to an ORP of 209.4
calibration value = 209.4
209.4 x 10 = 2094
2094 to HEX = 0x0000082E

calibration MSB Register = 0x00
calibration high byte Register = 0x00
calibration low byte Register = 0x08
calibration LSB Register = 0x2E



0x0C – Calibration request register

1 unsigned byte

Command values
1 = Clear calibration (delete all calibration data)
2 = Single point calibration

By default this register will read 0x00. When a calibration request command has been sent and a stop command has been issued, the ORP OEM™ will perform that calibration requested. Once the calibration has been done the Calibration Request Registers value will return to 0x00.

0x0D – Calibration confirmation register

1 unsigned byte

Command values
0 = no calibration
1 = calibration

After a calibration event has been successfully carried out, the calibration confirmation register will reflect what that calibration has been done.

Settings to this register are retained if the power is cut.

Sensor data

0x0E: ORP reading MSB	R/W
0x0F: ORP reading high byte	R/W
0x10: ORP reading low byte	R/W
0x11: ORP reading LSB	R/W

0x0E – 0x11 ORP reading registers

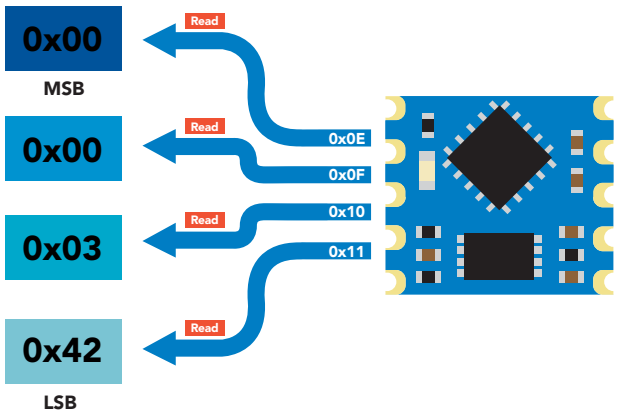
Signed long
0x0E = MSB
0x11 = LSB
Units = mV

The last ORP reading taken is stored in these four registers. To read the value in this register, read the bytes MSB to LSB and assign them to a signed long, cast to a float. Divide that number by 10.

Example

Reading an ORP of 83.4 mV

Step 1 read 4 bytes



Step 2 read unsigned long

Step 3 cast unsigned long to a float

Step 4 divide by 10

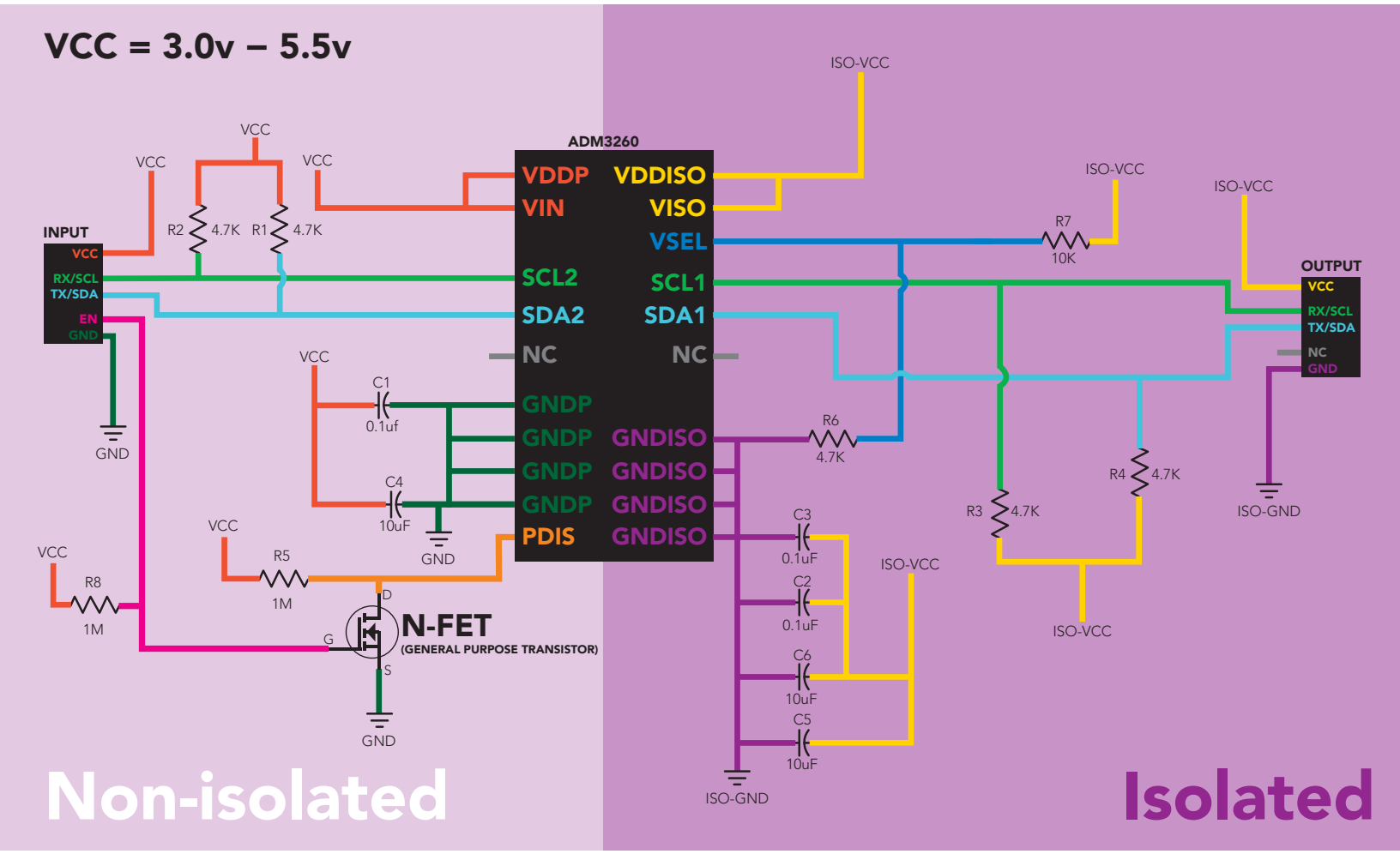
OEM electrical isolation

If the ORP OEM™ Class Embedded Circuit is going to be used in consumer, industrial, or scientific/medical applications electrical isolation is strongly recommended. Electrically isolating the device will insure that the readings are accurate, the ORP probe does not interfere with other sensors and that outside electrical noise does not affect the device.

The goal of electrically isolating the ORP OEM™ device is to insure that the device no longer shares a common ground with the master CPU, other sensors and other devices that are can be traced back to a common ground. It is important to keep in mind that simply isolating the power and ground is not enough. Both data lines (SDA, SCL) and the INT pin must also be isolated.

This technology works by using tiny transformers to induce the voltage across an air gap. PCB layout requires special attention for EMI/EMC and RF Control, having proper ground planes and keeping the capacitors as close to the chip as possible are crucial for proper performance. The two data channels have a 4.7kΩ pull up resistor on both the isolated and non-isolated lines (R1, R2, R3, and R4) The output voltage is set using a voltage divider (R6 and R7) this produces a voltage of 3.9V regardless of your input voltage.

Isolated ground is different from non-isolated ground, these two lines should not be connected together.

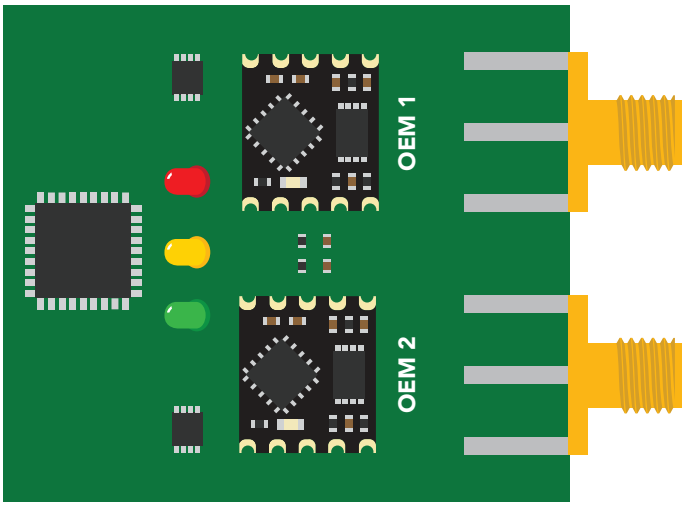


Designing your product

The ORP OEM™ circuit is a sensitive device. Special care ***MUST*** be taken to ensure your ORP readings are accurate.

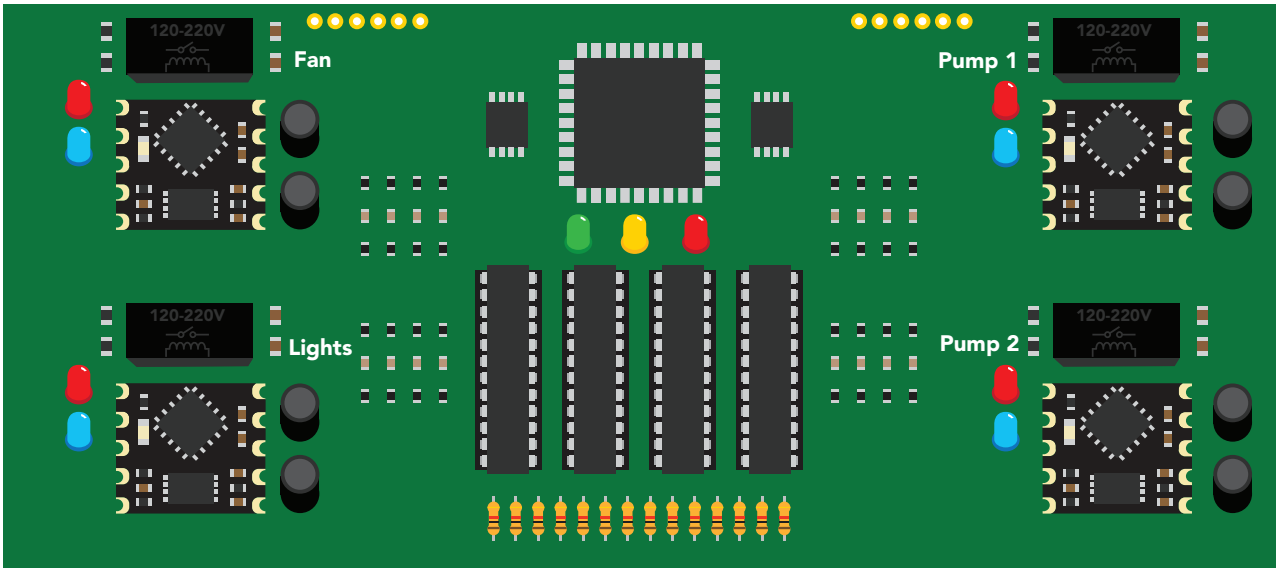
Simple design

Simple low voltage computer systems experience little to no problems during development and have no reported issues from the target customer.



Complex design

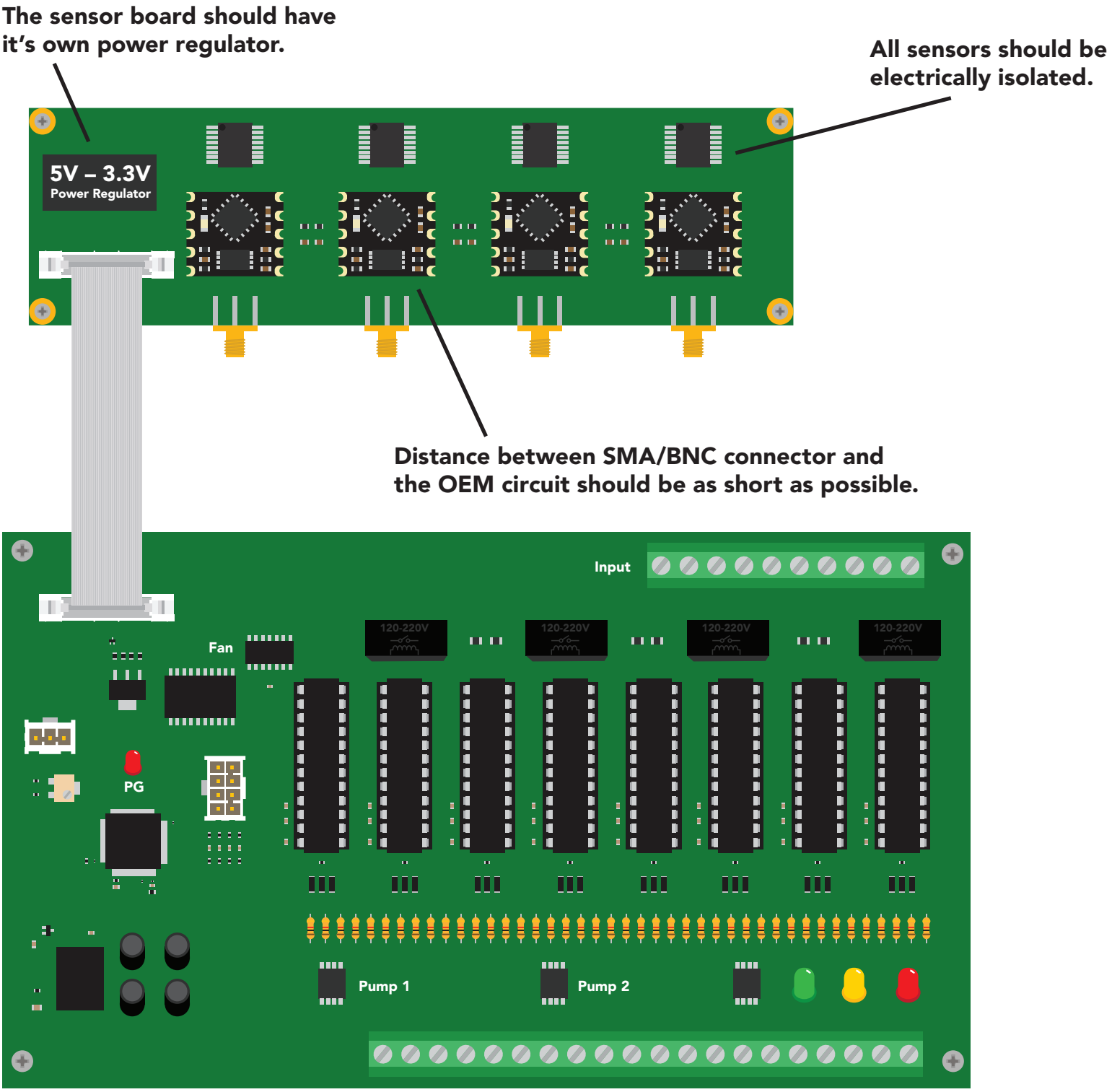
Complex computer systems with multiple voltages and switching, can lead to extended and unnecessary debugging time. Target customers can experience frequent accuracy issues.



How to add chemical sensing to a complex computer system

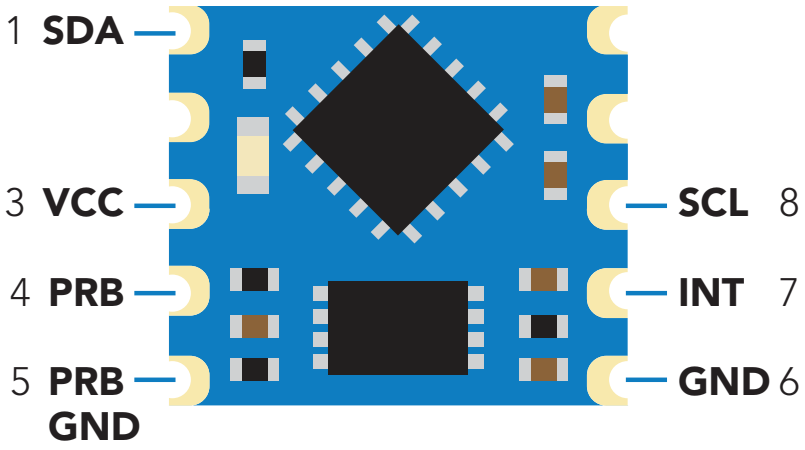
Placing the OEM™ circuits onto their own board is ***strongly recommended***; Not only does this help keep the design layout simple and easy to follow, it also significantly reduces debugging and development time.

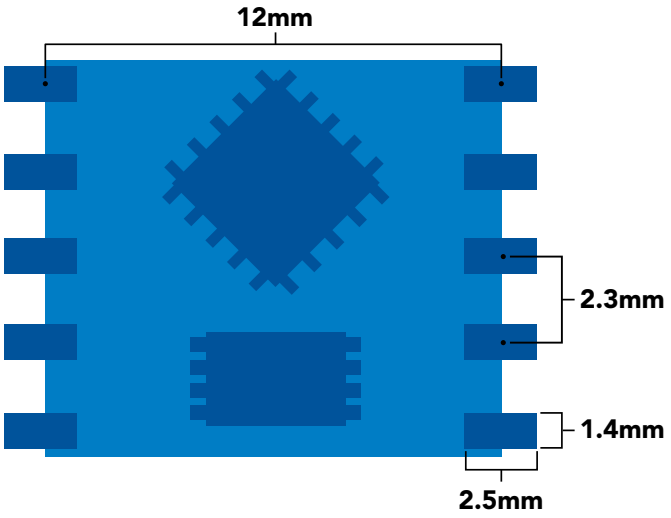
Target customers will experience accurate, stable and repeatable readings for the life of your product.



AtlasScientific™ Environmental Robotics

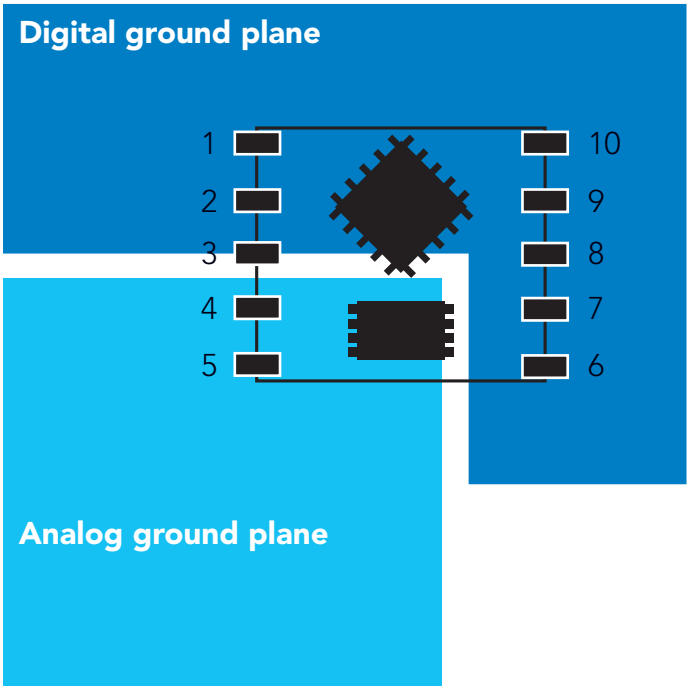
Designing your PCB



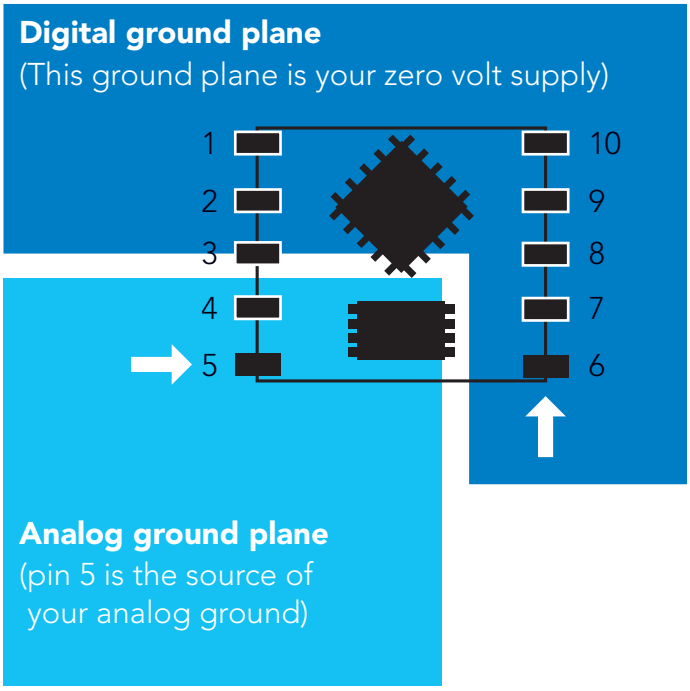


The ORP OEM™ circuit requires two separate ground planes to operate properly.
One ground plane is for the digital section of the device, the other is for the analog section.

1 Create two double-sided ground planes, just like the image below.

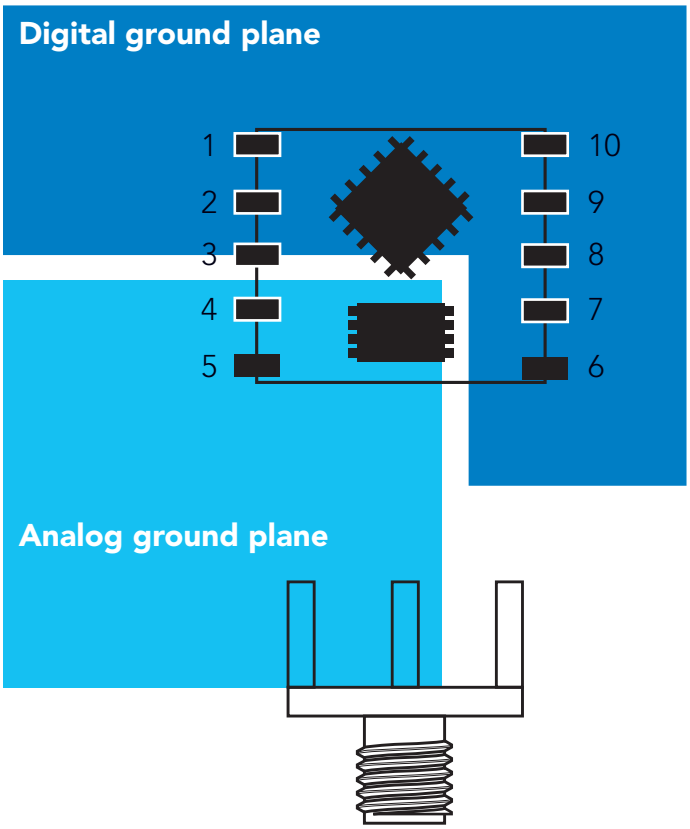


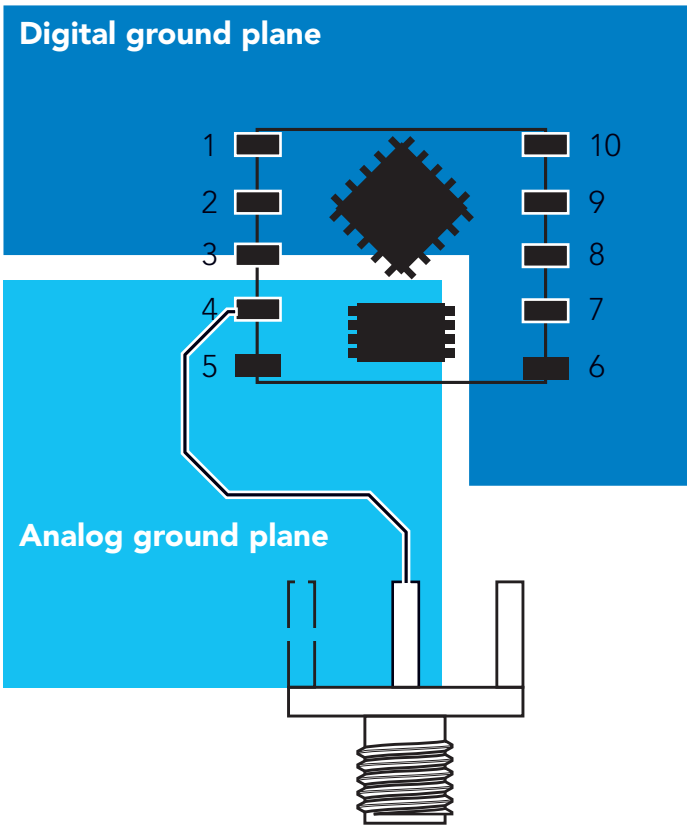
2 Connect pin 5 to the analog ground plane, and pin 6 to the digital ground plane.



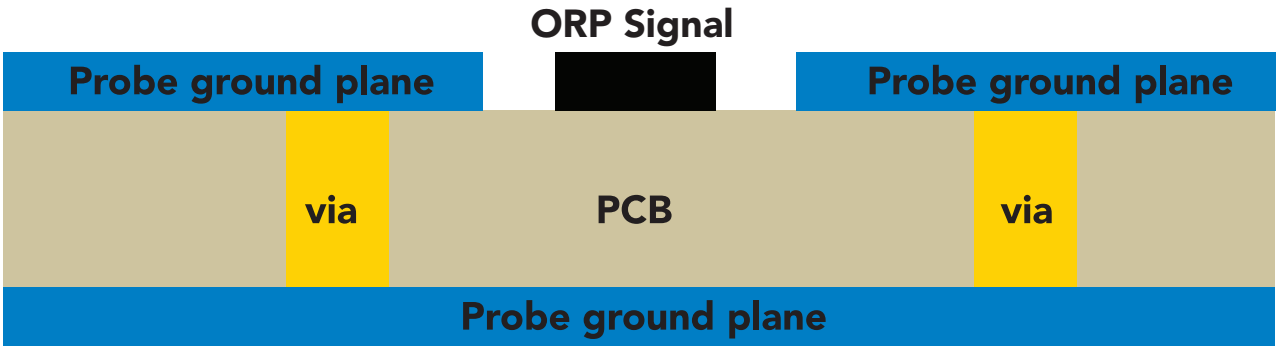
3 Place the probe connector (BNC/SMA) close to the ORP OEM™ circuit.

4 Using a 0.4mm trace width connect pin 4 (PRB) to pin 1 on the BNC/SMA. Keep this trace as short as possible. This trace is the ORP signal path.



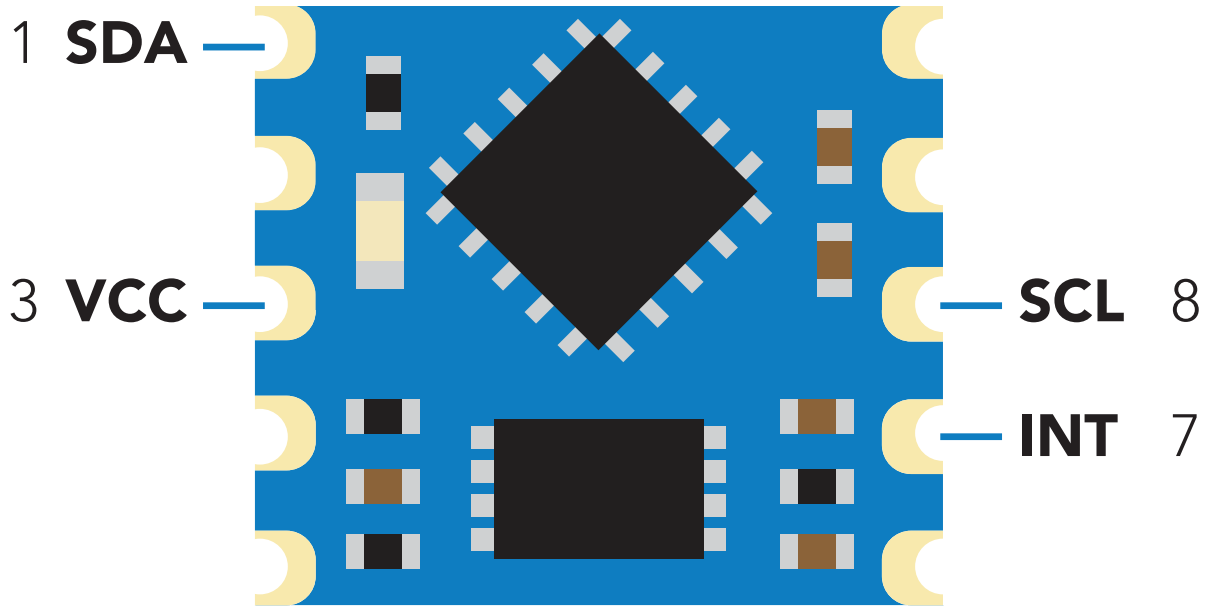


Cross section of the ORP signal path

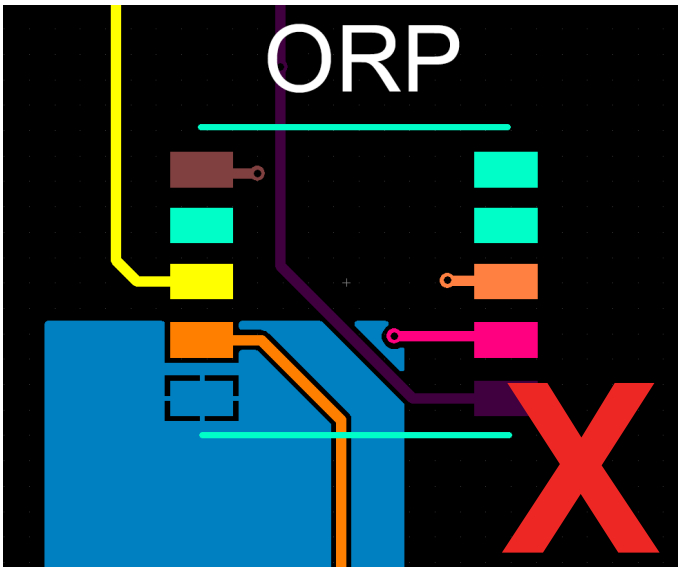


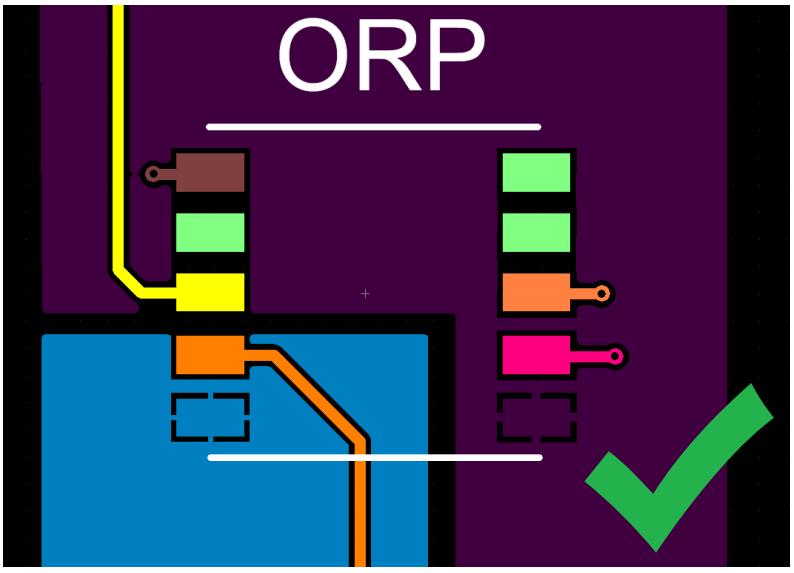
This cross section is an example of how the analog ground plane protects the ORP signal. The analog ground should surround the ORP signal, on both the top and bottom layers.

5 Rout the other traces as you see fit. If pin 7(INT) is unused leave it floating, *do not connect pin 7 to VCC or ground.*

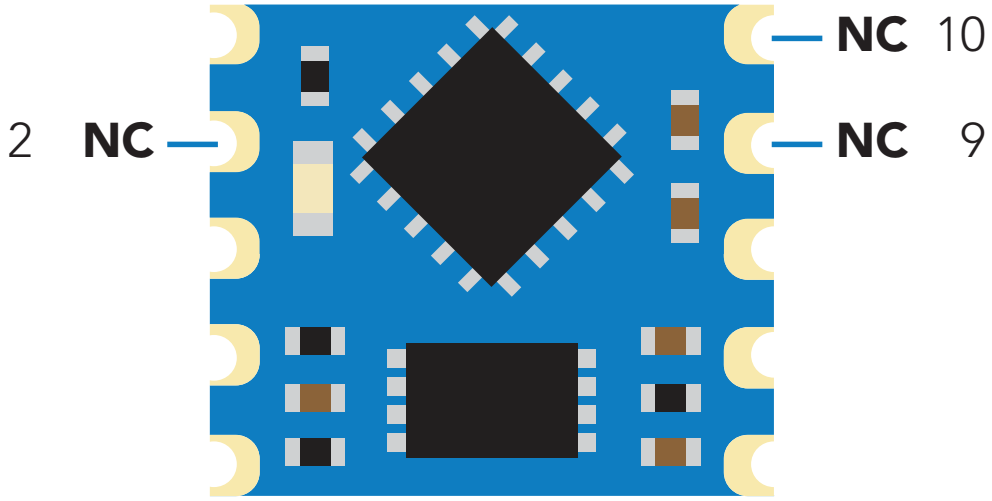


NEVER place vias under the OEM footprint.

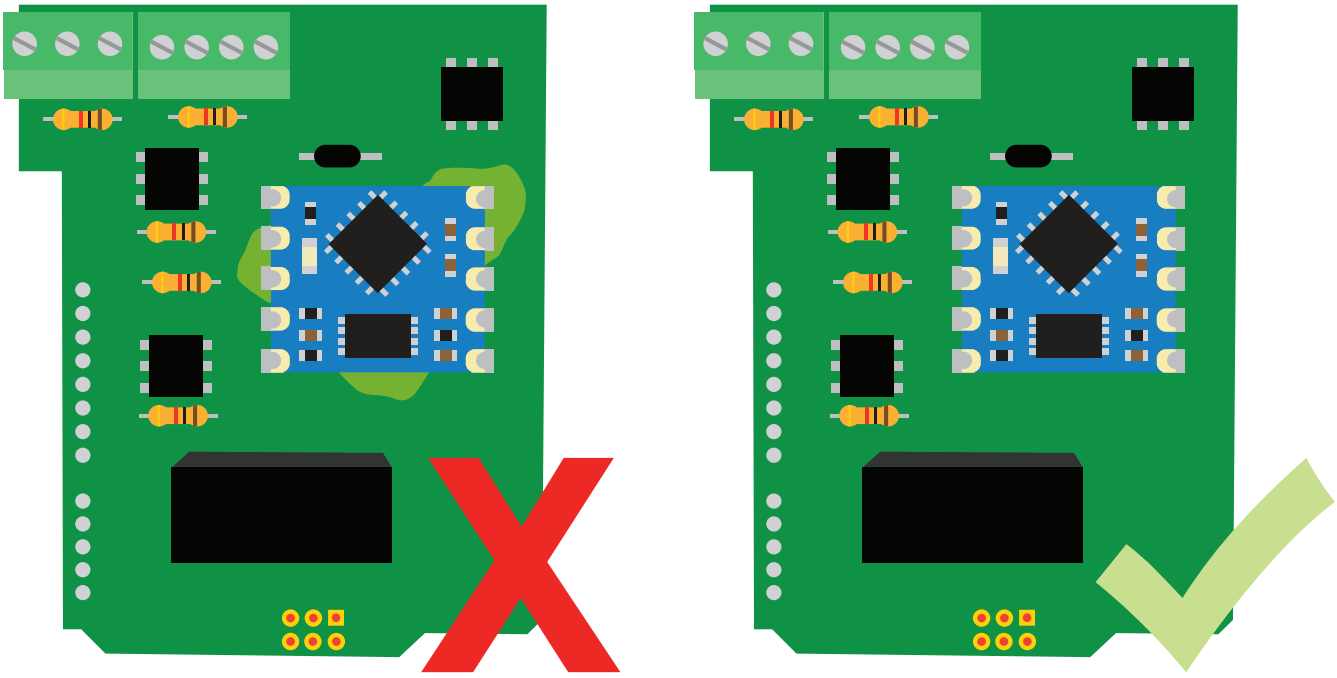




6 Pins marked NC *(No Connect)* must be left floating. ***NEVER*** *connect pins marked NC to VCC or ground.*



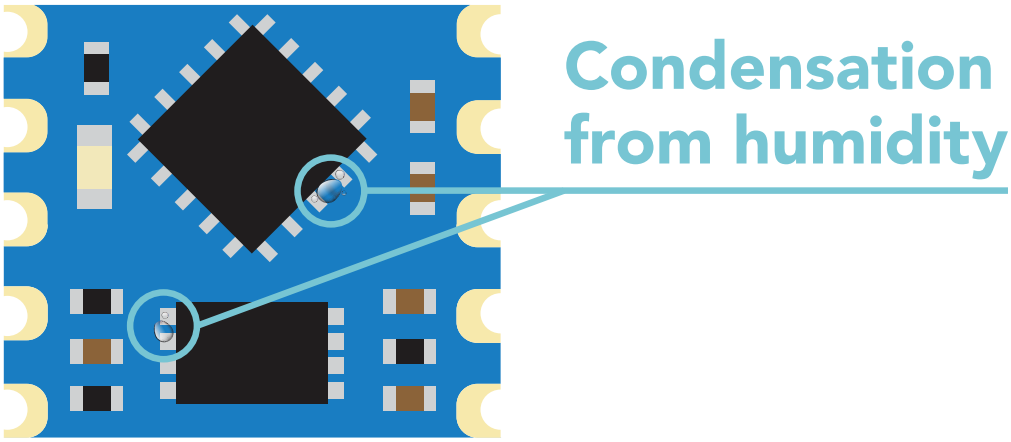
7 If the ORP OEM™ circuit is going to be hand soldered, avoid using rosin core solder. Use as little flux as possible. Do not let liquid flux seep under the ORP OEM™ circuit. After the ORP OEM™ circuit has been soldered to the PCB all flux residue *MUST* be removed. Failure to do so will result in poor quality readings.



DO NOT SKIP THIS STEP
The PCB must be washed with an ultrasonic cleaner, OR cleaned with a commercial flux removing chemical, OR soaked in alcohol for ~20 minutes.

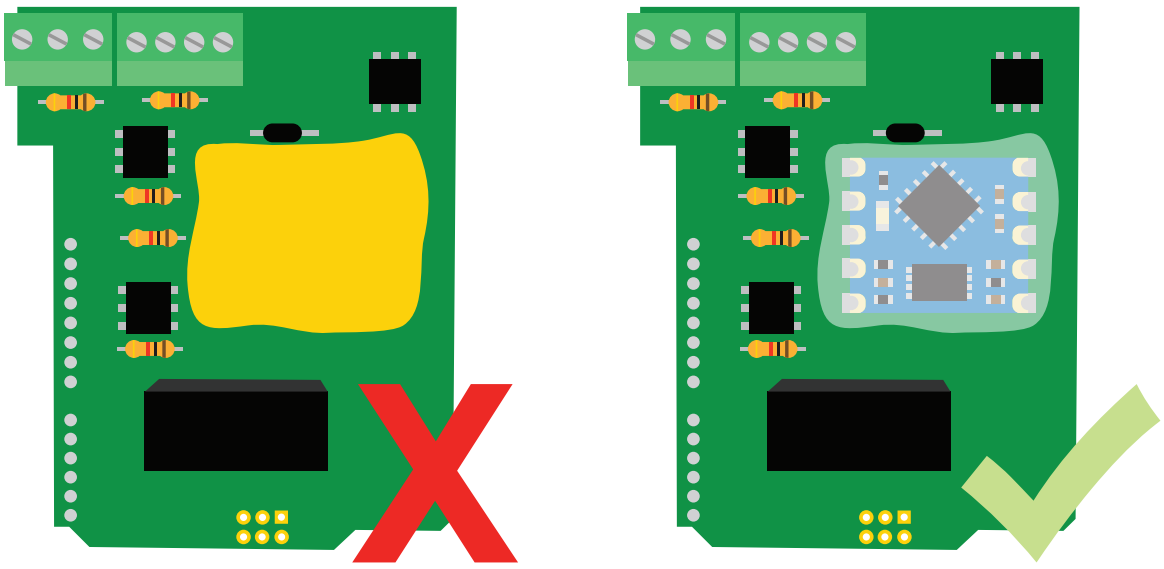
Humidity shielding

Humidity levels ***above 65%*** can affect the ORP readings. If you believe the target customer will operate the equipment in a humid environment, act accordingly.



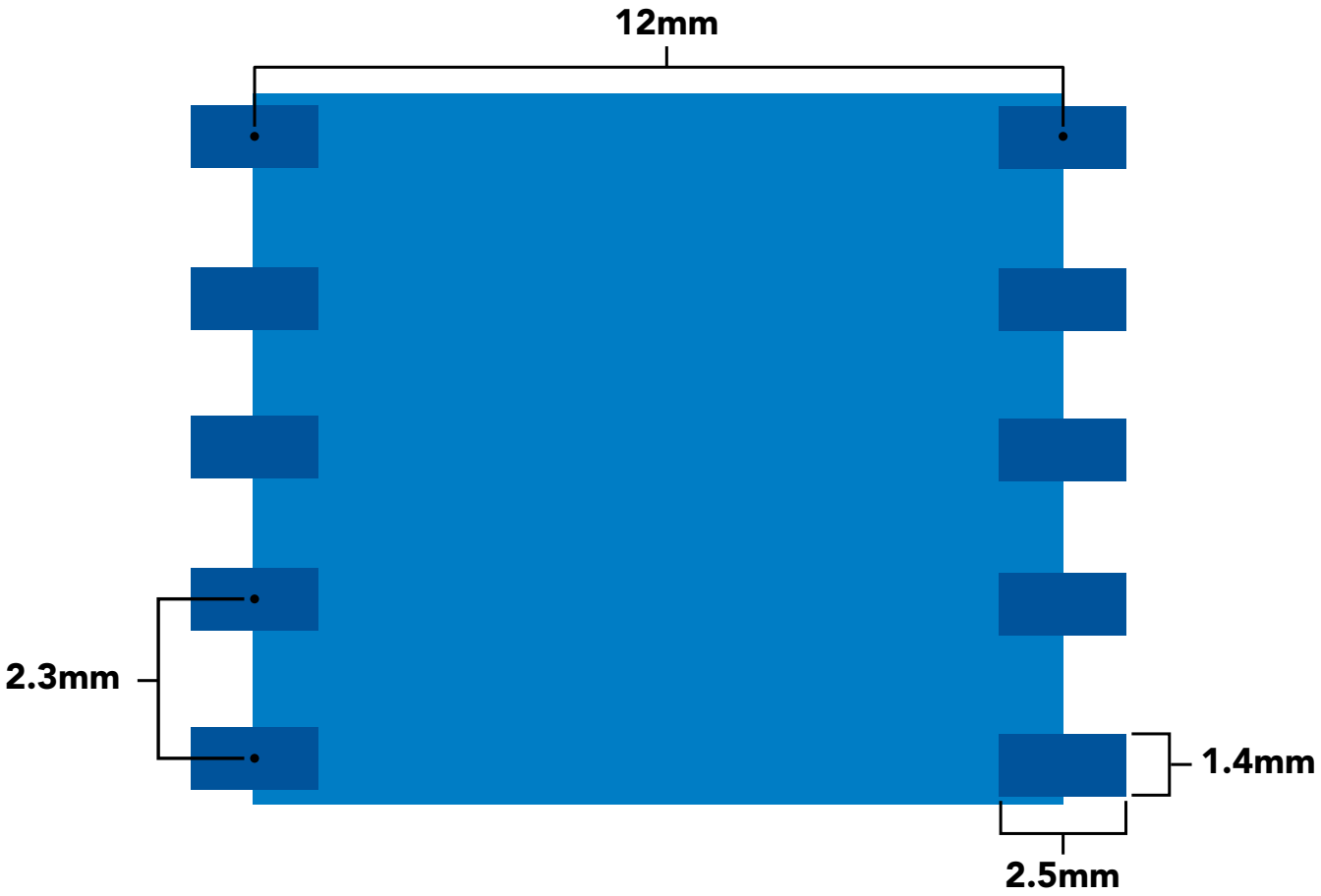
Shielding process:

1. Solder the OEM-ORP circuit to your PCB.
2. Wash the board to ensure that all flux has been removed.
3. Test to ensure the OEM-ORP circuit is delivering accurate ORP readings.
4. Encase the OEM-ORP circuit in ***clear epoxy***.



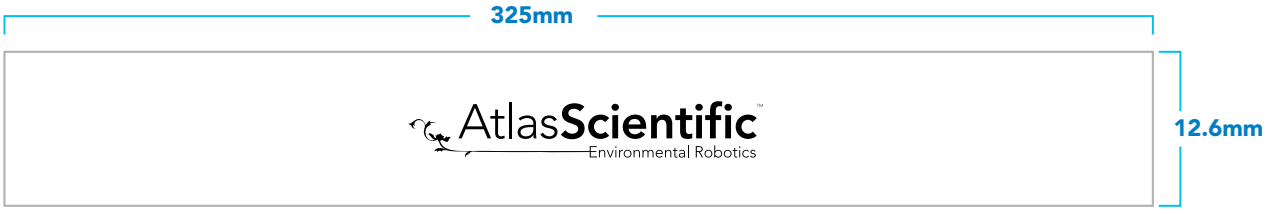
Only Use Clear Epoxy!
Do not use colored epoxy. Do not use black potting compound designed for electronics. Using colored epoxy or black potting compound will severely damage the OEM-ORP circuit.

Recommended pad layout

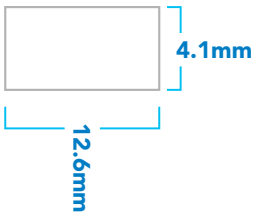


IC tube measurements

Top View



Side View

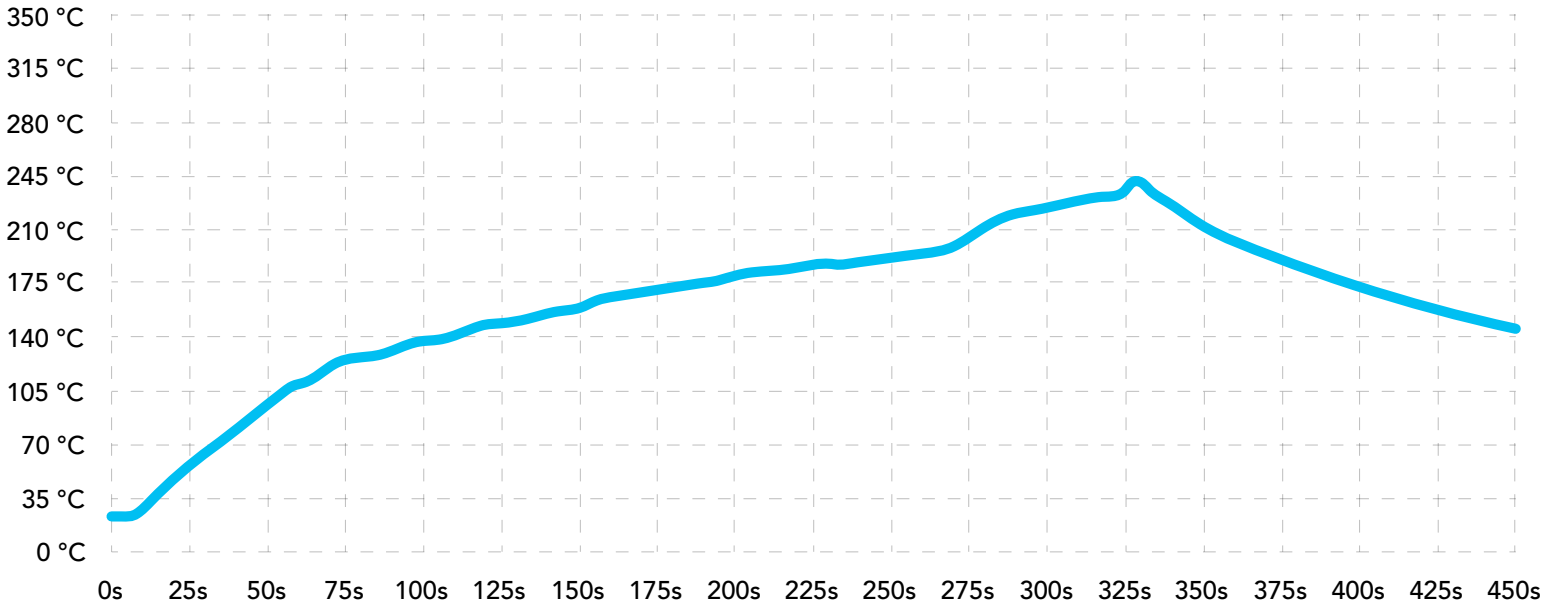


Fits 25 ORP OEM™ circuits

	inside dimensions	outside dimensions
L	325mm	325mm
W	11.6mm	12.6mm
H	3.1mm	4.1mm

plastic thickness 0.5mm

Recommended reflow soldering profile



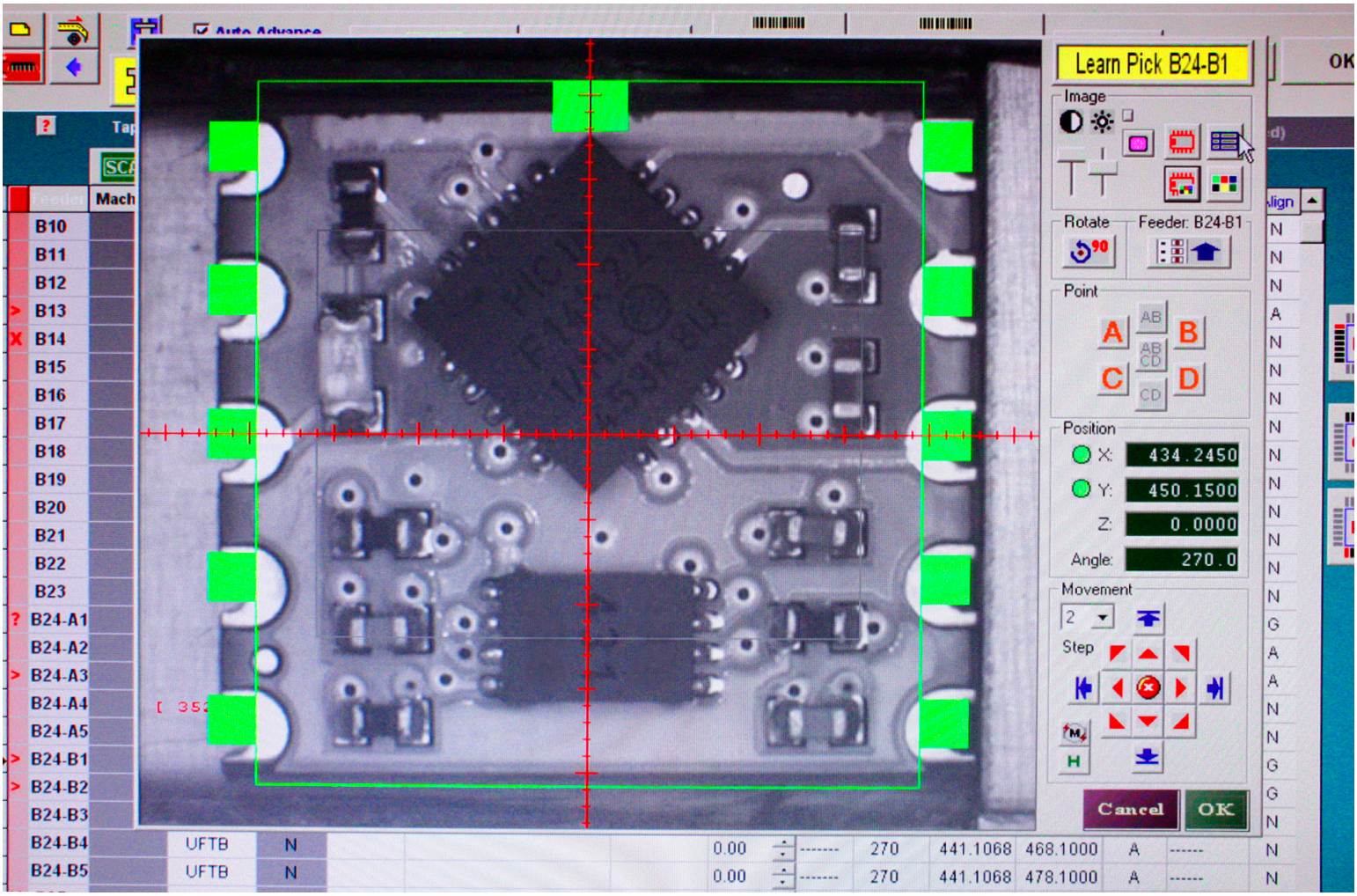
#	Temp	Sec
1	30	15
2	90	20
3	110	8
4	130	5
5	135	5
6	140	5
7	155	8
8	156	10
9	158	10
10	160	10

#	Temp	Sec
11	163	10
12	165	10
13	167	10
14	170	10
15	172	10
16	174	10
17	176	10
18	178	10
19	180	10
20	181	10

#	Temp	Sec
21	182	10
22	183	10
23	185	10
24	187	10
25	220	30
26	225	20
27	230	20
28	235	8
29	170	20
30	130	20

#	Temp	Sec
31	100	25
32	80	30
33	30	30
34	0	15

Pick and place usage



Datasheet change log

Datasheet V 3.1

Added information on humidity and the OEM circuit on pg 30.

Datasheet V 3.0

Added new graphic on pg 3.

Datasheet V 2.9

Revised operating voltages on pages 1 & 4.

Datasheet V 2.8

Revised artwork on pg 7.

Datasheet V 2.7

Added "Designing you product" on pg 23.

Datasheet V 2.6

Changed Calibration Confrim register 0x0D from R/W to R.

Datasheet V 2.5

Expanded upon the "Designing your PCB" section of datasheet, pg. 23

Datasheet V 2.4

Firmware update.

Datasheet V 2.3

Revised isolation schematic on pg. 22

Datasheet V 2.2

Changed "Max rate" to "Response time" on cover page.

Datasheet V 2.1

Corrected max rate reading on cover page.

Datasheet V 2.0

Revised entire datasheet

Firmware updates

V4.0 – Initial release (Feb 14, 2016)

V5.0 – (November 27, 2018)

- Fixed a bug where the calibration status didn't load correctly on power up.

